Implementing Air Conditioner Controller between CC3200 with OLED Display and Android Application

Wenbo Geng
Computer Engineering
University of California Davis
wgeng@ucdavis.edu

Pu Sun
Computer Engineering
University of California Davis
psun@ucdavis.edu

***Abstract*—We are implementing a air conditioner control by developing a Android application with a CC3200 Launch Pad. We made this design out of the transformation of the living environment. In this article, we will describe in detail how we use thermometers, WIFI, OLED screens, SPI protocol, I2C protocol, Android applications and AWS services to achieve two-way interaction. UART can be optionally used for debugging. The final product can control the air conditioner through the mobile phone and the cc3200, and transfer the data to each other using AWS.**

***Index Terms*—embedded system, temperature sensor, OLED, API Gateway**

I. Introduction

The CC3200 Launch Pad and OLED used for UC Davis EEC 172 instruction can be used to implement a air conditioner controller using the temperature sensor built in to CC3200 can be used as main input, the OLED is used as main display of the selection menu and temperature display. The two push buttons in CC3200 are used in the selection control to adjust the temperature or switching the temperature unit like Fahrenheit or centigrade in the menu. The CC3200 also provides WiFi capability, which allows us to do basic communication with the IoT services and restful API for temperature control with the Android application etc.

II. Motivation

Our ideas is come from the Google Nest Learning Thermostat. Using a application to control the air conditioner base on most of the family are not using intelligent household electrical appliance. If people want to adjust the temperature for the air conditioner, they have to adjust from the controller and it's not remotely. So base on that we deiced to make a application to control the temperature to the CC3200 and update to the AWS to make two way communication. We will also incorporate knowledge from UI design and visualization to optimize the air conditioner control.

III. Approach

This project uses the code from EEC 172 labs as templates. Our goal for this project is to implement a controller that can control the heat or cooling system for the air conditioner depending on the temperature changing. So at the beginning is to have a proof of concept work showing that all components used are functional and there are no pin conflicts when using all of them together and the temperature sensor is working correctly. The other goal is to display the temperature on the OLED. And connected with IoT things. After that the goal is to make a Android application that can connect to AWS to implement the two way communication which the user can control the air conditioner through their application. At the end optimizate our design like adding a menu selections.

IV. Components Used

All components used in this project are provided in EEC 172 Lab. Specifically,

- CC3200 Launch Pad [1]
  - Temperature Sensor [2]
  - WiFi module
  - UART port
  - Push buttons
  - Slow Clock [3]
- OEL Display Module (OLED) [4]

A. Communication Protocols

The CC3200 CPU communicates with the temperature sensor using I2C Protocol.

The CC3200 CPU communicates with the OLED using the SPI protocol.

AWS Amplify(Restful) and API Gateway (Restful) updating shadows

B. Pin Mux Configuration

Pin Mux configuration is done using the TI PinMux tool.

- GPIO
  - GPIO13: Pin 4, input, for SW3
  - GPIO22: Pin 15, input, for SW2
  - GPIO28: Pin 18, output, for Cool Control
  - GPIO31: Pin 45, output, for Heat Control
  - GPIO07: Pin 62, output, for OLED Reset
  - GPIO08: Pin 63, output, for OLED OC
  - GPIO09: Pin 64, output, for OLED DC

- I^2C
  - SCL: Pin 01, for I^2C SCL
  - SDA: Pin 02, for I^2C SDA
- SPI
  - CLK: Pin 5, for OLED CL
  - MOSI: Pin 7, for OLED SI
- UART
  - RX: Pin 57, for UART
  - TX: Pin 55, for UART

C. Jumper Wire Connections

Extra connections need to be made between the CC3200 Launch Pad and the OLED. 7 Jumper wires are required in total. See Table I for details.

TABLE I
JUMPER WIRE CONNECTIONS

CC3200		**OLED**	**Ref**
Pin	**Pin**	**Pin**	
VCC	–	Vin	–
GND	–	GND	–
P05	–	CL	–
P07	–	SI	–
P18	–	–	Cool Control
P45	–	–	Heat Control
P62	–	–	Reset
P63	–	OC	–
P64	–	DC	–
J2	J2	–	SCL
J3	J3	–	SDA

V. DESIGN LOGIC

Our design is basically divided into two parts: css code controls the cc3200 microprocessor, and kotlin code controls the Android APP. The basic logic in css is implemented in the `main.c` file. All button functions are implemented in `MainActivity` in Android Studio, and the basic property settings of all buttons are completed in the `activity_main.xml` file. The API upload endpoint and secret key were changed in the other json files, and the icons were updated in the other xml files.

1) The overall logic is implemented in the `main()` function. This function continuously loops in `while(1)`, reading the state of the two buttons. Then press the button to enter the setting or switch the cooling and heating of the air conditioner. Then the equation will wait for a while. The temperature sensor will re-read the temperature every time the cycle is executed 100 times. This is to ensure that the temperature will not keep uploading new data to aws. If the temperature changes, the temperature drop is uploaded to aws and displayed on the OLED screen.
2) Because we have updated the display function provided in the test, after the setting is selected, the OLED screen can directly display the setting interface and enter the `setting()` function. In this function, we can further choose to set the desired air conditioner temperature or change the temperature. This will also enter two different functions to set the temperature or adjust the temperature unit to Fahrenheit or Celsius.
3) Using the `controlAC()` function, we can make pin45 output 1 when the temperature is lower than the set value when the warm air is turned on, and output 1 when the temperature is higher than the set temperature when the cold air is on. In other cases, the output is 0. According to the output of the central air conditioning controller we disassembled, we can control the relay to connect to the air conditioner in this way.But because we did not order a suitable relay, we did not connect to the air conditioner display in the demo.
4) The `http_post()` function in lab4 is split into two functions `http_post()` and `http_get()` to facilitate uploading or updating data to cc3200 at the right time.
5) In the Android kotlin code, the function is implemented under each different button. The `Temp+` and `Temp-` buttons can adjust the set temperature, the `ON/OFF` switch can control the air conditioner switch, the `heat/cool` switch controls the cold and hot air, and the unit switch controls the temperature unit. But these data will not be transmitted to AWS immediately, unless the `post` button is pressed. When the `get` button is pressed, the latest shadow data will be updated on the mobile phone.

VI. OPTIMIZATIONS

A. Analysis Provided Graphics Library on OLED Design

An analysis to the provided graphics library in EEC 172 Lab 2 shows that there are a number of inefficiencies. One example is using `drawChar()` and `drawCircle()` to give the user that having a selection menu and `drawCircle()` is used in the button for user to selected and `drawChar()` is used in print out the message for the user.

B. Analysis on Android Studio UI Design

Using the Android Studio features like the button and snow and sun iron to make our application more visualized which makes the user earlier to understand the feature in the application and how to use it.

VII. EXTRA FEATURES

A. Two Way Communication AWS Amplifier (Restful API)

The temperature sensor communicates with Amazon AWS service for storing the temperature value and user's setting in the shadows . Base on the AWS API-Gateway can directly update the shadow to the thing we made for CC3200 board and use the API endpoint + the private key for our call phone to connect the application to the CC3200 board. Then use the HTTP post method to keep updating the temperature value which is sending the current temperature, the temperature that user set for the air conditioner and whether it's heat, cool or off to the AWS. Since the temperature sensor is updating to fast which updates the temperature per seconds. So we delay the temperature sensor detection. And use `sprintf()` , `strcpy()` and `strcat()` to send the temperature value to the HTTP response.

It uses HTTP Get method to get the current temperature, user set temperature for the air conditioner and from AWS since we get the temperature whether it's heat, cool or off from the AWS is a char array which is a string value so use forced conversion which (char[] - 0) * 10 to convert the string value to a integer value

As a result, the user can see the setting both on the Android Application and OLED.

B. Screen Rotation for Android Application

Using the automaticly setting on the Android studio we implement a screen rotation for Android application.

VIII. Reproducing

Here are the procedures to reproduce this work

A. Requirements

- CC3200 Launch Pad (CC3200-LAUNCHXL)
  - USB cable
- Adafruit 1.5" SSD1351 128x128 RGB OLED
- 7 jumper wires
- A WiFi access point
- An Amazon AWS account
- Software cc3200: Composer Studio, UniFlash, and pinmux
- Android phone(oneplus 6 for demo)
- Software Android: Android Studio and npm

B. Wire Connections

- Connect the OLED screen with the CC3200 Launch Pad. See Table I.
- Connect the CC3200 to a computer using the USB cable.

C. Setting Up AWS

1) Go to AWS IoT [5], create a Thing. Create certificates for it and activate it. Also download Amazon AWS's Root CA.
2) Copy the REST API Endpoint host name from the Interact tab of the thing.
3) Go to IoT Policies and add a policy for allowing `iot:GetThingShadow` and `iot:UpdateThingShadow`.
4) Attach this policy to the certificates you have generated.
5) Use OpenSSL to convert the certificates to der format.
6) For android part, download by using npm
7) Follow the tutorial to complete the setting in the terminal.

D. Updating Code

Update all things mentioned in lab4 and adding new code to `main.c`. Android code is starting with android SDK and build for our entire app.

E. Flashing Program

The flashing configuration is saved in `project.usf`. You need to change `/sys/mcuing.bin` to use the project binary, `/cert/rootCA.der`, `/cert/client.der`, `/cert/private.der` to use the files from Section VIII-C. After you are done click the "Program" button.

IX. Future Work

- Updating the data which the temperature we read and user's setting in to the Amazon S3 bucket which can store the data
- Having more UI Design features on our Andriod application like having more images or iron
- Allow the user set by times like 7pm to start heating at temperature 76 Fahrenheit.
- Connect with IDO and relay to the air conditioner to test our experiment.

X. Conclusion

This project implements a air conditioner control which user can adjust the temperature for the air conditioner though their cell phone. The temperature reading is used in CC3200 build in temperature sensor and connected through I^2C the OLED connected through SPI, the WiFi module, the UART port, Slow Clock and AWS API Gateway. It contain our design logic, temperature detection and two way communication between our Android application and CC3200 board which included our highly optimized code in order to make two way communication. It also contains screen rotation feature for our Android application.

Acknowledgment

I would like to thank the UC Davis Electrical and Computer Engineering Department for providing equipments (CC3200 Launch Pad, OLED) for this project. I would also like to thank Professor Homayoun and EEC 172 TAs especially our TA Ruijie Fang for their instructions and helps.

References

[1] Texas Instruments, "CC3200 SimpleLink Wi-Fi and Internet of Things Solution With MCU LaunchPad Hardware User's Guide," June 2014, Revised March 2020.

[2] Bosch Sensortec, "BMA 222 Digital, triaxial acceleration sensor Data sheet," May 2012.

[3] Texas Instruments, "CC3200 SimpleLink Wi-Fi and Internet-of-Things Solution, a Single Chip Wireless MCU Technical Reference Manual," June 2014, Revised May 2018.

[4] Univision Technology Inc, "OEL Display Module Product Specification," October 2008.

[5] Amazon Web Services, "AWS IoT Developer Guide," April 2020.

[6] Amplify, "Amplify Libraries" June 2021.